Remarks

Claims 1-17 are currently pending in this application. Claims 1-12 are original, Claims 13 and 14 are currently amended, and Claims 15-17 are new.

Claims 1, 2, 4-6 and 9-13 are rejected under 35 U.S.C. 102(b) as being anticipated by Gostanian et al. (US Pat. 5,781,910).

Regarding Claim 1.

Claim 1 recites:

1. A method for executing two or more computational operations upon elements of a data structure, the method comprising the steps of:
(a) determining if any of the two or more computational operations to be executed are operable upon a same element;
(b) determining if any of the two or more computational operations determined to be operable upon the same element are in kind operations;
(c) determining if any of the two or more computational operations determined to be operable upon the same element and to be in kind operations are addition or assignment operations; and
(d) executing the two or more computational operations determined to be operable upon the same element, to be in kind operations, and to be addition operations.

In rejecting Claim 1 the Examiner states:

Gostanian discloses a method in which it is determined that two or more operations are operable on the same element and that the operations are in kind operations (column 10, line 15-column 11, line 10 of Gostanian). It is then determined that the operations are addition operations, thus commutative (column 10, line 15-column 11, line 10 of Gostanian).

In response to the Applicant's argument that the cited art does not teach "*determining if any of the two or more computational operations to be executed are operable upon a same element*," the Examiner states in the current office action that "Gostanian determines whether or not a specific operation and all of the other operations that might be performed on the element are commutative, thus operable on the same element in its current state (column 10, line 34-column 11, line 10 of Gostanian)." The Applicant disagrees with the Examiner's conclusion that the cited text teaches "*determining if ... operations to be executed are operable upon a same element*." There does not appear to be any discussion in Gostanian regarding a determination of whether or not operations operate on the same element. In the Examiner Interview of July 7, 2005 the Examiner specifically pointed out two examples in Column 10 of Gostanian and suggested that these examples provided sufficient support for the Examiner's position. The Applicant traverses this suggestion. The first example, at lines 41-43, includes "if the only transaction allowed by the system is a transaction that adds some amount (e.g., $500.00) to a deposit account, then these transactions are commutative." The second example, at lines 54-59, includes "For example, the transaction described above would become non-commutative if the system also supported a second transaction that reads the value of a deposit account, multiplies that value by some percentage (e.g., 10%) and adds this amount back to the initial value of the deposit account." The Examiner apparently interprets these two examples as teaching "*determining if any of the two or more computational operations ... are operable upon a same element*," because of the language "a deposit account."

The Applicant believes that this interpretation is incorrect for at least the following reasons:

(A) Contrary to the Examiner's position, any determination regarding commutativity that is made in Gostanian appears to be based on transaction types, without regard to which elements the transactions actually operate upon. This position is supported by the explicit teaching "[t]ransactions may be commutative or non-commutative depending on the type of transaction and on the other possible transactions that might be requested by the application clients," (Col. 10 lines 34-37).

(B) In the first example of column 10, the transactions are commutative because they are all addition operations, rather than because they operate upon "a deposit account." It is well known that addition operations are normally commutative regardless of what data elements the addition operations operate on. There does not appear to be any teaching within <u>Gostanian</u> to suggest that commutativity was based on anything other than this well known property of addition, much less that commutativity was based on the transactions having been determined to operate upon the same element. That commutativity, in <u>Gostanian</u>, is based on merely transactions types is again supported by the teaching of <u>Gostanian</u>, pointed out above, that states, "[t]ransactions may be commutative or non-commutative depending on the type of transaction and on the other possible transactions that might be requested by the application clients."

In further support of the Applicant's position, the difference between the first example, where the transactions are found to be commutative, and the second example, where the transactions are found to be non-commutative, is that the transactions are of two different particular types. The two examples do **not** differ regarding whether the transactions operate on the same element or not. In both examples, the transactions operate on "a deposit account." Thus, in these examples, the determination of commutativity is not taught to be dependent on whether or not the transactions operate on the same element. Rather the determination of commutativity is taught to be dependent on the types of transactions supported. This teaches away from the Examiner's suggestion that, in Gostanian, commutativity is based on "*determining if ... computational operations ... are operable upon a same element*." The Applicant is unable to find any teaching within Gostanian that whether or not transactions operate on the same element is used to determine whether the transactions are commutative or not. Thus, there would be no reason for Gostanian to perform a step of "*determining if ... computational operations ... are operable upon a same element*."

(C) The Applicant believes that the Examiner's conclusion that the two cited examples teach "***determining if** ... computational operations ... are operable upon a same element*," is incorrect. In both Gostanian examples, transactions operate on "a deposit account." It is the Applicant's position that, given such a single deposit account scenario, there would be no need to determine **if** two transactions operate on the same element, because they do so by definition. Thus, in the two examples of Col. 10, there would be no need for a step of "*determining if ... computational operations ... are operable upon a same element*," as recited in Claim 1. Therefore, transactions operating on "a deposit account", as taught in Gostanian, does not teach a step of "*determining if ... computational operations ... are operable upon a same element*."

(D) The Applicant believes that the Examiner's interpretation of Gostanian is incorrect because the examples of Col. 10 are consistent with the overall teachings of Gostanian without any need to infer an actual determination that transactions operate on the same deposit accounts. Thus, there is no reason to infer a step of "*determining if ... computational operations ... are operable upon a same element*" in the teachings of Gostanian. In the first example, including only addition transactions, all the transactions would still be commutative regardless of how many elements they operated on. In the second example, including addition and multiplication, a result that the transactions are non-commutative would be satisfactory and consistent with the rest of the teachings of Gostanian. In both examples, results are reached without a need for considering whether transactions are on the same element or on different elements. Thus, the examples in Col 10 do not imply a step of "*determining if ... computational operations ... are operable upon a same element*," as recited in Claim 1.

Furthermore, it is the position of the Applicant that Gostanian does not teach the limitation "*operations to be executed*" as recited in "*method for executing two or more computational operations ... comprising the steps of: (a) determining if any of the two or more computational **operations to be executed** are operable upon a same element...* ," of Claim 1. This limitation is not taught because Gostanian determines commutativity based on all the transactions "supported", rather than on the basis of "*operations to be executed*," recited in Claim 1. For example, at Col. 10 lines 55-56, it is taught that the commutativity of a transaction changes "if the system also supported a second transaction."

A determination based on all transactions supported is clearly distinct from a determination based on "*operations to be executed.*" Those operations that are "*to be executed*" at any particular time is typically a subset of all the operations supported, and may be a much smaller number of operations. Further, whether or not operations are commutative is only an issue when those operations are "*to be executed.*" By considering all supported operations, as is taught in Gostanian, transactions may unnecessarily be considered non-commutative. Thus, there is a substantial difference between determining commutativity based on supported transactions rather than on those operations actually to be executed.

For example, in Gostanian a system that supports two addition operations and a multiplication operation would **always** be non-commutative. In contrast, the method of Claim 1 could find commutativity in the same system if only the two additions operations were being executed at a particular time. This difference may be important in those typical systems that support a range of operations only a few of which are "*to be executed*" at any particular time. The Applicant is unable to find any teaching within Gostanian that all supported transactions are actually to be executed at a particular time. Thus, Gostanian does not teach a "*method for executing two or more computational operations ... comprising the steps of: (a) determining if any of the two or more computational operations to be executed are operable upon a same element... .*"

Furthermore, the Applicant maintains his position that Gostanian does not teach "*...determining if any of two or more computational operations ...* ***are in kind operations***," as recited in Claim 1. In response to the Applicant's argument that Gostanian does not teach these limitations, the Examiner refers to Col. 10 of Gostanian at lines 41-44 and 54-59 and states "Gostanian shows examples in which this determination is used directly to select the protocol. In the first case, the system determines the operations are all addition (in kind), thus commutative..." The Applicant disagrees with the Examiner's interpretation of Gostanian.

It is the position of the Applicant that the two examples given do not constitute an "in kind" determination. While the first example (lines 41-44) happens to include all addition operations, as discussed above the operations are determined to be commutative because they are of a particular type of operation, e.g., addition operations, not because they are all of the same type of operation. For example, in the system of <u>Gostanian</u> the first example would still be commutative if addition and subtraction operations were supported. In this case, the particular types of transactions, e.g., addition and subtraction, would be used to find that the transactions were commutative, despite there being two different kinds of transactions.

It is not possible that the test for commutativity used in <u>Gostanian</u> is that "in kind" transactions are commutative while transactions that are not "in kind" are non-commutative. For example, if all operations were assignment operations they would be of the same kind but not necessarily commutative. Thus, the Examiner's suggestion that "Gostanian shows examples in which this determination is used directly to select the protocol," is incorrect. The interpretation of <u>Gostanian</u> suggested by the Examiner is unworkable because it would fail to identify non-commutative operations. In the examples of <u>Gostanian</u>, commutativity is determined based on the specific types of transactions supported, not based on whether they are "in kind" or not.

Further, if it were assumed, for the sake of argument, that <u>Gostanian</u> used an "in kind" test for commutativity, the systems of <u>Gostanian</u> would be non-functional. Because, as pointed out above, the determination of commutativity in <u>Gostanian</u> is based on the entire set of operations *supported* rather than a subset of those operations *actually to be executed*, any functional system that supported more than one different type of transaction would never include non-commutative transactions. Thus, in typical systems that include more than one type of transaction, the use of an "in kind" rule, applied to all supported transactions, would be non-functional. Therefore, Gostanian does not include an "in kind" rule and does not teach "*(b) determining if any of the two or more computational operations determined to be operable upon the same element are in kind operations*" as recited in Claim 1.

In summary, it is the Applicant's position that Gostanian does not teach several of the limitations of Claim 1, among other possible differences. These limitations include, for example, "*determining if ... computational operations ... are operable upon a same element,*" "*determining if any of the two or more computational* ***operations to be executed*** *are operable on the same element...,*" and "... *determining if any of two or more computational operations determined to be operable upon the same element are* ***in kind operations***." For at least the above reasons, the Applicant believes that Claim 1 is allowable.

Regarding Claim 2.

Claim 2 recites:

2. The method of claim 1 further comprising the steps of:
(e) determining, of the two or more computational operations determined to be operable upon the same element, to be in kind operations, and to be assignment operations, if a same value is to be assigned to the same element; and
(f) executing the two or more computational operations determined to be operable upon the same element, to be in kind operations, to be assignment operations, and to assign the same value to the same element.

In rejecting Claim 2, the Examiner first suggests that Gostanian teaches a testing mechanism for determining commutativity of transactions. The Examiner then suggests that the existence of this testing mechanism implies "*(e) determining, ... if a same value* *is to be assigned to the same element*," as recited in Claim 2. The Applicant traverses both of these steps in the Examiner's reasoning.

Specifically, in rejecting Claim 2, the Examiner states on page 9 of the current office action:

> Gostanian teaches that a determination of protocol is made based on a determination of whether or not transactions on data are commutative or non-commutative (column 10, lines 33-37 of Gostanian). This determination is made by testing how data would be effected by the transactions being executed in different orders, if the final state is the same they are commutative, if it is not they are not commulative (column 10, lines 33-50 of Gostanian).

First, the Applicant respectfully points out that this statement appears to be contradictory to the Examiner's positions in the discussion of Claim 1, in which the Examiner suggested that the determination of commutativity was base on a determination that operations were "in kind" or operating on the same element, rather than performing an actual test. For example, if the determination of commutativity were, for the sake of argument, assumed to be based on actual testing of how data is affected by executing operations in different orders, then there would be no reason for determining that operations are "in kind" and Claim 1 would be allowable.

Second, the Applicant disagrees with the suggestion that the determination of commutativity is made by actually testing how data would be affected by the transactions being executed in different orders. Such an approach is clearly impractical when it is remembered that the object of <u>Gostanian</u> is to gain efficiency by using a 1PCC protocol instead of a 2PCC protocol when possible, i.e., when operations are commutative. Each test suggested by the Examiner would, itself, be non-commutative and would require use of the 2PCC protocol. Thus, the approach suggested by the Examiner would involve executing four operations using the 2PCC protocol merely to determine if the 1PCC could be used instead of the 2PCC. This would clearly eliminate any efficiency and, thus, be counter to the teachings of Gostanian.

In the Examiner Interview of July 7, 2005, the Applicant's undersigned representative requested that the Examiner specifically point out a teaching within the cited art of "testing how data would be affected by the transactions being executed in different orders, if the final state is the same they are commutative," as the Examiner suggests is taught. In response, the Examiner pointed to Col. 10 lines 51-54 and Col. 11 lines 4-10. The Applicant disagrees with the suggestion that these texts teach an actual test to determine commutativity.

The text at Col. 10 lines 51-54 is a definition of a non-commutative transaction. Specifically, the text includes "[a] transaction is non-commutative if the final state of the database server is dependent on [t]he order in which this transaction is executed..." The text does not say that commutatively is determined by actually trying the transaction in different orders, as suggested by the Examiner.

The text at Col. 11 lines 4-10 includes:

> If a particular replicated database is not able to execute the transaction in the same order as the other databases (e.g., another transaction has a lock on the relevant deposit account number), then as described below the 2PCC protocol causes the transaction to be aborted in each replicated database and the system waits before trying to execute the transaction again.

This text concerns individual steps during utilization of the 2PCC protocol after the commutativity of transactions has **already been determined** to be non-commutative, and does not concern a test for commutativity. This is shown by the immediately preceding text which states "[i]n this situation [non-commutative transactions], the system must utilize the 2PCC protocol described in detail below serializing the execution of these transactions and thus ensuring that all replicated database servers reach the same final result." Further, the failure discussed in the cited text is a failure of a database to execute a transaction in a particular order, not a failure to arrive at the same result after executing the transaction is different orders.

Third, on page 10 of the current office action, the Examiner states:

> By definition two transactions that are assignment transactions on the same element with the same assignment value (for instance: transaction 1 is A=3 and transaction 2 is A=3) when tested would result in an identical final states, thus be deemed commutative when operated in the method of Gostanian, and be allowed to execute under the open protocol (column 10 lines 37-41 of Gostanian).

The Applicant understands this statement by the Examiner to indicate that the Examiner believes that because <u>Gostanian</u> teaches the above suggested test for commutativity, <u>Gostanian</u> teaches "*(e) determining, ... if a same value is to be assigned to the same element*," as recited in Claim 2. However, even if for the sake of argument, one were to assume that <u>Gostanian</u> taught the test suggested by the Examiner, such a test would not teach "*(e) determining, ... if a same value is to be assigned to the same element.*" There is no determination of whether the same value is assigned in the actual test suggested by the Examiner. Further, while two assignments assigning the same value would be commutative, the converse is not necessarily true. Two assignments may be commutative while assigning different values. Thus, a finding that two transactions are commutative does not mean that they assign the same value. The transactions could be commutative for other reasons, e.g., because the transactions are on different data elements. Thus, a finding that transactions are commutative is not sufficient to determine "*if a same value is to be assigned*," as recited in Claim 2.

For at least the above reasons, the Applicant believes that the cited art does not teach "*(e) determining, of the two or more computational operations determined to be operable upon the same element, to be in kind operations, and to be assignment operations, if a same value is to be assigned to the same element*," and, therefore, Claim 2 is allowable.

The Applicant further believes that Claim 2 is allowable for at least the reasons discussed with respect to Claim 1, from which it depends.

Regarding Claims 4-6, 9 and 11.

It is the Applicant's position that Claims 4-6, 9 and 11 are allowable for at least the reasons discussed above with respect to Claim 1.

Regarding Claim 10.

It is the Applicant's position that Claim 10 is allowable for at least the reasons discussed above with respect to Claims 1 and 2.

Regarding Claim 12.

Claim 12 recites:

12. A method for executing two computational operations upon elements of a data structure, the method comprising the steps of:
executing the two computational operations if
either computational operation does not violate a limit, and
both computational operations do not operate upon a same element;
executing the two computational operations if
either computational operation does not violate the limit,
both computational operations operate upon the same element, and
both computational operations are addition operations; and
executing the computational operations if

> *either computational operation does not violate the limit,*
> *both computational operations operate upon the same element, and*
> *both computational operations are assignment operations that assign a same value to the same element.*

In the current office action, the Examiner states "Applicant's ... arguments regarding 'limits' are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection has been made and is shown above." The Applicant respectfully points out that the new grounds for rejection differ from the previous grounds only with respect to those claim elements involving limit violation. The Examiner has, thus, failed to address the Applicant's previous arguments regarding "*executing the two computational operations if ... both computational operations do not operate upon a same element,*" and "*executing the computational operations if ... and both computational operations are assignment operations that assign a same value to the same element,*" as recited in Claim 12. The Applicant requests that the Examiner address these arguments, repeated below, or allow Claim 12.

It continues to be the Applicant's position that the cited art does not teach, "*executing the two computational operations if ... both computational operations do not operate upon a same element,*" as recited in Claim 12. In rejecting Claim 12, the Examiner previously stated "[i]n one case, the operations will be executed if they are not operational on the same elements (column 10, line 15 through column 11, line 10 of Gostanian)." The Applicant is unable to find any such teaching within the cited text. Specifically, there does not appear to be any teaching that commutativity of an operation is determined responsive to a conditional determination as to whether another operation operates on the same element. This point was discussed above with respect to Claim 1.

The Applicant therefore, respectfully requests that the Examiner more particularly point out this teaching within the Gostanian or allow Claim 12.

Furthermore, it continues to be the Applicant's position that the cited art does not teach, "*executing the computational operations if ... and both computational operations are assignment operations that assign a same value to the same element*," as recited in Claim 12. In rejecting Claim 12, the Examiner further stated:

> In yet another case they will execute if they operate on the same element and they are assigning the same value to the element, based on the fact that Gostanian discloses a method in which it is determined that the result of two operations is listed as commutative and thus are allowed to execute if it is determined that the result would be identical if the operations were run in any order, which would include two identical assignment statements. (Column 10, line 15-column 11, line 10 of Gostanian).

The Applicant is unable to find any such teaching within the cited text. Specifically, while Claim 12 recites a conditional limitation using the work "*if*," there does not appear to be any conditional aspect taught in Gostanian wherein execution of an operation is dependant on a condition that either of two or more operations violate a limit, that both operations operate on the same element, ***or*** that both operations include assigning a same value.

For at least these reasons, and those reasons discussed herein with respect to Claims 1 and 2, the Applicant believes that Claim 12 is in condition for allowance.

Regarding Claims 13-14.

Claims 13-14 have been amended in response to the Examiner Interview of July 7, 2005. Claim 13, as amended, recites executing two computation operations if one but not the other of the computational operations violates a limit. Claim 14, as amended, recites executing two computational operations if one or none of the operations violates a limit but not if both of the two computational operations violate a limit.

It is the Applicant's position that these amendments place Claims 13 and 14 in condition for allowance.

Further, the Applicant believes that Claims 13 and 14 are allowable for the same reasons discussed above with respect to Claim 12.

Claims 3, 7 and 8 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gostanian in view of Chen et al. (U.S. Patent Number 4,901,230).

Regarding Claim 3.

Claim 3 recites:

3. The method of claim 2 further comprising the step of:
determining if any of the two or more computational operations determined to be operable upon the same element and to be in kind operations violate a limit, then not performing steps (d) or (f).

The Applicant believes that Claim 3 is allowable for at least the reasons discussed with respect to Claim 1 and Claim 2, from which it depends.

Regarding Claim 7.

Claim 7 recites:

7. A method for categorizing two or more computational operations executable upon elements of a data structure, the method comprising the steps of:
determining if any of the two or more computational operations violate a limit; and
categorizing the two or more computational operations determined to violate the limit as not commutative.

In rejecting Claim 7 the Examiner states "Chen discloses a method in which it is determined whether the computational operations violate a limit then the operations are considered to be a failure and not executed (column 17, lines 18-67, and column 3, lines 43-53). In the Examiner Interview of July 7, 2005, the Applicant's undersigned representative and the Examiner discussed the teachings of Chen and agreed that the teachings of Chen, as suggested above, do not teach "*categorizing the two or more computational operations determined to violate the limit as not commutative*," as recited in Claim 7. Specifically, Claim 7 recites that a limit violation results in a categorization of a specific type, and does not result in **failure** of the operation as is taught in Chen. The Examiner stated that he would review this rejection and the Applicant looks forward to receiving allowance of Claim 7, or a new ground for rejection.

Regarding Claim 8.

Claim 8 includes limitations similar to those recited in Claim 7. The Applicant, therefore, believes that Claim 8 is allowable for the same reasons discussed above with respect to Claim 7.

Regarding New Claims 15-17.

Claims 15-17 are supported by the same parts of the application as filed as are Claims 7 and 12 and are believed to be allowable for at least the same reasons as Claims 7 and 12.

The Applicant thanks the Examiner for the helpful Examiner Interview of July 7, 2005.

The Applicant believes that all pending claims are allowable and respectfully request that the Examiner issue a Notice of Allowance. Should the Examiner have questions, the Applicant's undersigned representative may be reached at the number provided below.

Respectfully submitted,
Clifford L. Hersh

Date: August 25, 2005

[illegible]

Steven M. Colby, Reg. No. 50,250
Carr & Ferrell *LLP*
2200 Geng Rd.
Palo Alto, CA 94303
Phone (650) 812-3424
Fax (650) 812-3444